REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-04-

0539

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE Final	3. DATES COVERED *(From - To)* 1 Oct 2002 - 30 Sep 2004

4. TITLE AND SUBTITLE Qualitive Detection of Independently Moving Targets in MPEG Video Within the Compressed Domain	5a. CONTRACT NUMBER
	5b. GRANT NUMBER F49620-03-1-0007
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Zhongfei Zhang	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Research Fund of State University of New York P. O. 6000 Binghamton, NY 13902-6000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research 4015 Wilson Blvd Mail Room 713 Arlington, VA 22203 Jm	10. SPONSOR/MONITOR'S ACRONYM(S) AFOSR
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release; distribution is unlimited.

20041028 084

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This is the final report to AFOSR and AFRL/IFEC for the project Qualitative
Detection of Independently Moving Targets in MPEG Video within the Compressed Domain. This project focuses on developing theory and techniques for large scale surveillance video data summarization and unsupervised segmentation of video streams regarding whether there is a presence of independent motion in the streams. We propose a holistic, in-compression approach to efficient video prostanding. By efficient, we mean that the processing speed is close to or even faster than real-time in "normal" platforms(we do not assume using special hardware or any parallel machines) while still maintaining comparable quality with state-of-the-art methods. By prostanding, we mean to aim at those tasks that are between the traditional video processing and traditional video understanding. We target surveillance applications.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Zhongfei Zhang
U	U	U	UU		19b. TELEPHONE NUMBER *(Include area code)*

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

EST AVAILABLE COPY

REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-04-

0539

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE Final	3. DATES COVERED *(From - To)* 1 Oct 2002 - 30 Sep 2004

4. TITLE AND SUBTITLE Qualitive Detection of Independently Moving Targets in MPEG Video Within the Compressed Domain	5a. CONTRACT NUMBER 5b. GRANT NUMBER F49620-03-1-0007 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Zhongfei Zhang	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Research Fund of State University of New York P. O. 6000 Binghamton, NY 13902-6000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research 4015 Wilson Blvd Mail Room 713 Arlington, VA 22203 NM	10. SPONSOR/MONITOR'S ACRONYM(S) AFOSR 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This is the final report to AFOSR and AFRL/IFEC for the project Qualitative
Detection of Independently Moving Targets in MPEG Video within the Compressed Domain. This project focuses on developing theory and techniques for large scale surveillance video data summarization and unsupervised segmentation of video streams regarding whether there is a presence of independent motion in the streams. We propose a holistic, in-compression approach to efficient video prostanding. By efficient, we mean that the processing speed is close to or even faster than real-time in "normal" platforms(we do not assume using special hardware or any parallel machines) while still maintaining comparable quality with state-of-the-art methods. By prostanding, we mean to aim at those tasks that are between the traditional video processing and traditional video understanding. We target surveillance applications.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Zhongfei Zhang
U	U	U	UU		19b. TELEPHONE NUMBER *(Include area code)*

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

A Holistic, In-Compression Approach to Video Prostanding

Final report submitted to AFOSR and AFRL/IFEC for the project *Qualitative Detection of Independently Moving Targets in MPEG Video within the Compressed Domain* **under the AFOSR grant F49620-03-1-0007**

PI: Zhongfei (Mark) Zhang
Computer Science Department
Watson School of Engineering and Applied Science
SUNY Binhamton
Binghamton, NY 13902 - 6000
Phone: (607) 777 2935
Fax: (607) 777 4729
Email: zhongfei@cs.binghamton.edu

Abstract

This is the final report to AFOSR and AFRL/IFEC for the project *Qualitative Detection of Independently Moving Targets in MPEG Video within the Compressed Domain.* This project focuses on developing theory and techniques for large scale surveillance video data summarization and unsupervised segmentation of video streams regarding whether there is a presence of independent motion in the streams. We propose a holistic, in-compression approach to efficient video prostanding. By efficient, we mean that the processing speed is close to or even faster than real-time in "normal" platforms (we do not assume using special hardware or any parallel machines) while still maintaining comparable quality with state-of-the-art methods. By prostanding, we mean to aim at those tasks that are between the traditional video processing and traditional video understanding. We target surveillance applications. Specifically, we focus on two prostanding tasks: independent motion detection and frame alignment. Solutions developed in the two prostanding tasks provide complementary roles in facilitating efficient browsing of a large collection of surveillance video: the independent motion detection tool allows identifying the shot with the "foreground" target motion without waiting for playing the whole collection of video before identifying the shots; the frame alignment tool allows accessing of the "background" scene immediately, through applications such as mosaicking, without waiting for playing the whole collection of video. For the independent motion detection task, we have developed the theory and the technique called **LSCA**. For the frame alignment task, we have developed the theory and the technique called **ICM**. Both techniques are representatives of the holistic, in-compression approach. Theoretical and experimental analyses show that both methods work robustly in solving their problems, and thus demonstrate and validate the holistic, in-compression approach in solving for video prostanding problems.

1 Introduction

This project focuses on developing theory and techniques for large scale surveillance video data summarization and unsupervised segmentation of video streams regarding whether there is a presence of independent motion in the streams. Since processing time is critical in summarizing and/or segmenting large scale surveillance video, the developed theory and techniques must provide efficient processing. Consequently, we focus on developing the theory and related methods aiming at efficient processing of MPEG video stream data. By *efficient*, we mean that the processing speed is close to or even faster than real-time in "normal" platforms (we do not assume using special hardware or any parallel machines) while still maintaining comparable quality with the state-of-the-art methods. Video prostanding incorporates research activities related to computer vision and image processing including video processing. We propose the concept of *video prostanding* to purposively focus on the activities that are between the *typical video processing* which aims at low-level processing of the video signals (such as compression, noise filtering, and frame based camera stabilization) and the *typical video understanding* which aims at semantic interpretation and understanding (such as scene recognition and activity analysis and interpretation).

We target surveillance applications. Based on the ultimate goal of efficient video prostanding, we propose the holistic, in-compression approach, and we demonstrate the approach by focusing on two video prostanding tasks: (1) independent motion detection; and (2) frame alignment. For the independent motion detection task, we have developed the theory and the technique called **LSCA**. For the frame alignment task, we have developed the theory and the technique called **ICM**. Solutions developed in the two prostanding tasks provide complementary roles in facilitating efficient browsing of a large collection of surveillance video: the independent motion detection tool allows identifying the shot with the "foreground" target motion without waiting for playing the whole collection of the video before identifying the shots; the frame alignment tool allows accessing of the "background" scene immediately, through applications such as mosaicking, without waiting for playing the whole collection of the video. Examples of the video prostanding applications are ubiquitous ranging from economic development to homeland security in which efficient prostanding of video streams is highly desirable.

This report is organized as follows. After this Introduction section, we give the motivation of this research as well as the related work in the literature in the next section. We then introduce the **LSCA** theory and the method and the **ICM** theory and the method in the following two sections, respectively. We then present extensive experimental data to demonstrate the promise and effectiveness of these methods. Finally, we conclude this report.

2 Motivations and Related Work

In motion detection, when the camera is still, the problem is relatively easy [19]. However, when a surveillance video is taken from a camera that is also in motion, every pixel in a frame may contain motion. For those background pixels, the motion reflected in the image domain corresponds to the 3D camera motion. On the other hand, for those pixels corresponding to independently moving objects in the frame, their motion corresponds to the combination of the 3D camera motion and their own independent motion in the 3D space. In this case, simple frame based differencing does not work [19], and certain sophisticated techniques must be applied to separate the independent motion from the camera motion, which is also called the background motion. This problem becomes even more complicated when there is 3D motion parallax involved. In this case, a 3D motion model must be applied in order to robustly and accurately separate the independent motion from the camera motion. Therefore, the problem of detection of independently moving objects here is reduced to the problem of independent motion detection. This is one of the problems we address in this project.

There are two scenarios related to independent motion detection. Given a video stream or an image sequence, one scenario refers to the detection in which a *temporal* segmentation is conducted into those subsequences (called shots) that contain the scene in which one or more independently moving objects are present, in addition to a *spatial* segmentation and delineation of each of the independently moving objects in each of the frames of these shots. The other scenario, on the other hand, refers to the detection in which only the *temporal* segmentation is conducted to return those shots that contain independent motion; no spatial segmentation is performed to identify the independently moving objects in each frame. The focus of this project is primarily in the latter approach.

Motion analysis has been a focused topic in computer vision and image understanding research for many years [46, 18, 15, 12, 30]. Due to the difficult nature of the problems in this topic, it is still considered as an open area and many research efforts are still being developed in this topic [10, 11]. Independent motion analysis, on the other hand, deals with multiple motion components simultaneously, and therefore, is presumably more challenging.

The earliest work in independent motion detection may be dated back to the early 80's. Jain [22] proposed a solution assuming that the camera was under a translation. Adiv [1] assumed the availability of optical flow and used the flow to group regions based on the rigidity constraint over two frames. Nelson [33] proposed two methods based on velocity constraints to detect independently moving objects. Thompson et al [44] used a similar approach based on the rigidity constraint. Bouthemy and Francois [9] treated the problem of independent motion detection as a statistical regularization problem and attempted to use the Markov Random Field model to solve for the problem. Ayer et al [7] used robust statistical regression techniques to detect independent motion. Smith and Brady [42] used geometric constraints for independent motion segmentation. Sharma and Aloimonos [39] provided a solution to this problem based on the normal flow field — the spatiotemporal derivatives of the image intensity function, as opposed to the typical optical flow field. Irani and Anandan [19] proposed a three-frames constraint based on a general 3D motion parallax model to detect independent motion. Argyros et al [5, 3, 4, 6] and Lourakis et al [27] used stereo camera streams to detect independent motion. Their techniques were essentially the combination of applying the normal flow field to the stereo streams and using robust statistical regression. Fejes and Davis [13] developed a low-dimensional, projection-based algorithm to separate independent motion using the epipolar structure of rigid 3D motion flow fields. Torr [45] proposed a method based on model selection and segmentation for separating multiple 3D motion components. Pless et al [35] provided a solution to the problem in a special case in which the scene may be approximated as a plane, which is valid for typical aerial surveillance. Their method is based on spatiotemporal intensity gradient measurements to directly compute an exact background motion model, and then the independent motion is detected based on the constraint violation for the mosaics developed over many frames. Sawhney et al [38] proposed a method that simultaneously exploits both constraints of epipolar and shape constancy over multiple frames. This method is based on the previous work on plane-plus-parallax decomposition [24, 37, 40], and thus requires explicitly estimating the epipolar and the homography between a pair of frames.

Most of the existing techniques for independent motion detection in the literature require spatial segmentation (i.e., identification) of the independently moving objects in the frames or images. Due to this fact, very few of them can afford fast detection (such as real time or even faster than real time detection), as their solutions to temporal independent motion detection depend on the spatial independent motion segmentations. While these approaches are useful in general, due to the specific applications that have motivated this project, in order to deliver a fast detection, we are only concerned with detecting those video shots that contain independent motion, without specifically identifying the independently moving objects in the frames. We argue that it is not necessary to identify the moving objects in the image frames in the applications that we are concerned with. This is based on the following two reasons. (i) In many applications, the time issue, i.e., the detection speed, is always an important concern. Obviously the spatial domain identification requires more processing time. (ii) It is not necessary to take the spatial domain identification approaches in many applications. Even if the independently moving objects are all segmented and identified in each frame, given the current status of computer vision and artificial intelligence in general, it is *not possible* to have a fully automated capability to interpret whether the segmented and identified independent motion in the frames indicates any specific significance without interaction with human expertise. Therefore, these detected shots must be sent to the users for further analysis anyway, *regardless* of whether or not the independently moving objects are segmented and identified in each frames in these shots.

The other observation is that in the literature, most of the existing techniques for independent motion detection are based on image sequences, as opposed to compressed video streams. In other words, given a video, such as a surveillance video, these methods require that the video must be first fully decompressed to recover an image sequence before these methods can be applied. This restriction (or assumption) significantly hinders these techniques from practical applications, as in today's world, information volume grows explosively, and all the video sequences are archived in compressed forms. This is particularly true in the surveillance applications this project is concerned with, in which the data volume is *massive* and they must be archived in a compressed form, such as MPEG.

Based on these considerations, we have developed a holistic, in-compression approach to solving for the problem of *efficient* independent motion detection *directly* from the compressed surveillance video streams. This capability allows two possible application scenarios for this technology. The first is to equip the sensors with this detection algorithm for real-time data scanning while the sensors are in surveillance. The second is to mine (or scan) an archived surveillance video database in which all the video data are stored in a compressed format to retrieve the shots containing independent motion.

While independent motion detection addresses detecting the "foreground" target in the surveillance video analysis, frame alignment facilitates applications such as video mosaicking to allow immediate access to the "background" scene. We primarily apply the frame alignment approach to video mosaicking in this project.

General methods of aligning frames starting from a sequence's pixels are quite mature, but more relevant to the discussion here are those methods that do processing in the *compressed domain*. Jones et at. [23] align video frames by fitting the received MPEG motion vectors to a three-parameter model, where the three parameters represent horizontal displacement, vertical displacement, and zoom; the parameters are estimated by averaging the received MPEG motion vectors. Milanese et al. [32] also use a three-parameter model to describe global camera motion, where the three parameters are estimated by minimizing a least-squares criterion. To help reduce the effect of motion-vector inaccuracies, the authors do two pre-processing steps where they apply spatial and temporal median filters to the motion vectors, followed by removing inconsistent motion vectors from the least-squares criterion. Pilu [34] uses a six-parameter affine model to describe the global motion between frames. A least-squares solution is found to match the received motion vectors, where similar to Milanese et al., some pre-processing is performed to reduce the impact of inaccurate motion vectors: the gradient of the image is thresholded to remove smooth image regions from consideration; the result is then median filtered, followed by a final linear smoothing. Meng and Chang [31] perform an affine fit to MPEG motion vectors. Rather than performing a pre-processing step to eliminate outlier motion vectors, the authors first fit all motion vectors to the affine model; motion vectors that have a very poor fit to the resulting model are subsequently removed, followed by a re-calculation of the model parameters. The process is repeated iteratively to refine the final results.

There are several limitations of the methods discussed above. Models that make use of only three parameters [23, 32] may work well for some problems, but are clearly inadequate for modeling motion more complex than translation and zoom. Pilu's method [34] uses an adequate six-parameter model, but the thresholding procedure for discarding motion vectors can be problematic—a threshold that works well for one sequence may be unsuitable for a different sequence. Meng and Chang's work [31] also uses adequate model complexity, but as discussed by the authors of the RANSAC algorithm [14], the general procedure of iterative model fitting followed by outlier rejection can lead to incorrect fits. There is also a fundamental problem with algorithms that use simple averaging or least-squares for fitting motion vectors to a global motion model: large outliers can severely corrupt the final result, due to the unnecessarily high quadratic penalty in a least-squares formulation. The frame-alignment algorithm presented in this report addresses this very topic.

3 LSCA Approach to Independent Motion Detection

We propose a holistic, in-compression approach to solving for the unsupervised segmentation problem for the independent motion detection based on the linear system consistency analysis theory, and thus, we call this approach as **LSCA**. Since the **LSCA** approach only focuses on what exactly is necessary to compute, it saves the computation to a minimum and achieves the efficacy to the maximum. Consequently, **LSCA** delivers an efficient solution to the independent motion detection task. In this section, we first describe the **LSCA** theory. Then we give the **LSCA** method. Finally, we give the derivation of the theoretic bound for detecting independent motion based on the **LSCA** method.

3.1 Linear System Consistency Analysis

We use a 3D to 2D affine model to approximate the video camera imaging system. For a typical surveillance video, this affine model is sufficiently accurate for the mapping from 3D scenes to 2D images. Our experiments also show that this model even works well for some of the non-surveillance video such as movies (see Fig. 6 for an example).

Given a 3D point $\boldsymbol{P}$ and its corresponding 2D point $\boldsymbol{p}$, a 3D to 2D affine transform is a linear transform, and is defined as [21]:

$$\boldsymbol{p} = \boldsymbol{A}\boldsymbol{P} + \boldsymbol{t} \tag{1}$$

where $\boldsymbol{A}$ is a 2 by 3 matrix with six independent parameters, and $\boldsymbol{t}$ is a 2D vector with another two independent parameters.

Assume that the camera motion between two arbitrary frames is an arbitrary 3D motion, which can be represented as a 3 by 3 rotation matrix $\boldsymbol{R}$ with three independent parameters, and a 3D translation vector $\boldsymbol{T}$ with another three independent parameters.

$$\boldsymbol{P}' = \boldsymbol{R}\boldsymbol{P} + \boldsymbol{T} \tag{2}$$

where $\boldsymbol{P}'$ is the same point of $\boldsymbol{P}$ after the camera motion in the 3D space. The displacement of the point $\boldsymbol{P}$ in the 3D space with respect to time after the motion is:

$$\dot{\boldsymbol{P}} = \boldsymbol{P}' - \boldsymbol{P} = (\boldsymbol{R} - \boldsymbol{I})\boldsymbol{P} + \boldsymbol{T} \tag{3}$$

where $\boldsymbol{I}$ is the identity matrix. From Eq. 1 and Eq. 3, it is clear:

$$\dot{\boldsymbol{p}} = \boldsymbol{A}\dot{\boldsymbol{P}} = \boldsymbol{A}(\boldsymbol{R} - \boldsymbol{I})\boldsymbol{P} + \boldsymbol{A}\boldsymbol{T} \tag{4}$$

Let $\boldsymbol{P} = (X, Y, Z)^T$ and $\boldsymbol{p} = (x, y)^T$. Given each image point $\boldsymbol{p}$, Eqs. 4 and 1 give rise to four independent equations. Eliminating $\boldsymbol{P}$, we obtain a linear constraint for each image point $\boldsymbol{p}$ in a video frame:

$$\dot{x} + \theta\dot{y} + \alpha x + \beta y + \gamma = 0 \tag{5}$$

where the variables α, β, γ, and θ are functions of the motion parameters $\boldsymbol{R}, \boldsymbol{T}$ between the two frames, and the sensor parameters $\boldsymbol{A}, \boldsymbol{t}$ with the following relationship:

$$\alpha = \frac{f(m_{ij})}{g(m_{ij})} \tag{6}$$

where m_{ij} are the motion parameters (the elements of $\boldsymbol{R}$ and $\boldsymbol{T}$) and/or the sensor parameters (the elements of $\boldsymbol{A}$ and $\boldsymbol{t}$), and f, g are both quadratic functions. Similar expressions exist for β, θ, γ.

When a pair of neighboring frames is determined, the motion parameters $\boldsymbol{R}, \boldsymbol{T}$ are constants for all the image points in the frames. We first assume that the sensor parameters $\boldsymbol{A}, \boldsymbol{t}$ are always constants. We will relax this assumption later. Hence, the variables α, β, γ, and θ are independent of the image coordinates x, y, and their derivatives

$\dot{x}, \dot{y}$ in Eq. 5. This indicates that for each point in a frame, there is a linear constraint represented in Eq. 5.

Now assume that we have n points identified in a frame, and that we know each point's displacement $(\dot{x}, \dot{y})^T$ to the next frame. Since each point contributes one linear constraint of Eq. 5, we have a linear system consisting of these n points:

$$D\xi = b \tag{7}$$

$$D = \begin{pmatrix} \dot{y_1} & x_1 & y_1 & 1 \\ \dots & & & \dots \\ \dot{y_n} & x_n & y_n & 1 \end{pmatrix} \quad \xi = \begin{pmatrix} \theta \\ \alpha \\ \beta \\ \gamma \end{pmatrix} \quad b = \begin{pmatrix} -\dot{x_1} \\ \dots \\ -\dot{x_n} \end{pmatrix}$$

Given such a linear system, if we know that all the n points are with the camera motion, i.e., there is no independent motion with any of these n points, then all the n points have the same motion parameters. Since the sensor parameters are always the same for all the points, the n points will have the same values $\alpha, \beta, \theta, \gamma$ based on Eq. 6. Therefore, the linear system Eq. 7 is consistent, i.e., there are solutions to this system. This has proven the following theorem:

Theorem 3.1 *Given n points represented in the linear system of Eq. 7, if there is no independent motion with any of these points, then the linear system is consistent.*

This means that the consistency of the linear system is the necessary condition of no independent motion in the n points. In general, given $n > 4$, the rank of D is 4. Consequently, in general the consistency of Eq. 7 means there is a unique solution to this linear system.

From Theorem 3.1, it is clear that if the linear system Eq. 7 is not consistent, there must be independent motion involved. However, the linear consistency of the system Eq. 7 is not the sufficient condition for detecting any independent motion of the n points. This is due to the fact that Eq. 6 is not a one to one mapping between a specific vector of ξ and a specific set of motion parameters. Given the same solution ξ for all the n points, it is possible to have different values of motion parameters that satisfy the same mapping of Eq. 6. Fig. 1 illustrates one example of this situation. In this example, the camera moves from the position O at time t_1 to the position O' at time t_2 with a different orientation. If there is a 3D point Q which does not involve any independent motion between t_1 and t_2, then the corresponding images of Q at these two instants are p and q, respectively. Thus, the displacement between the two frames for Q in the image domain is $q - p$. Now as another scenario, if there is another 3D point P located somewhere in the ray between Q and O at time t_1, then the image of P is the same p as that of Q. When the camera moves from O to O', P undergoes an independent motion to Q. Consequently, the combined motion of P results in the same displacement vector $q - p$ in the image domain. This means that given the same displacement vectors, together with the same points in one frame, subsequently resulting in the same ξ vector from Eq. 7, we cannot tell whether there is independent motion involved. Hence, we have disproved that the linear system consistency is the sufficient condition for detecting independent motion.

Though the consistency of the linear system Eq. 7 is only the necessary condition and not the sufficient condition to determine whether there is any independent motion involved in the n point set, we can still use it to detect the independent motion. In

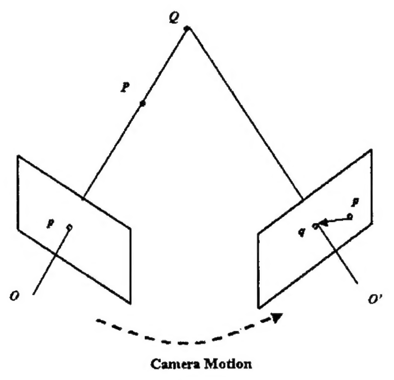


Figure 1: An example to show that the same displacement, which generates the same $\boldsymbol{\xi}$ vector, is not sufficient to tell whether there is an independent motion.

fact, if the linear system is not consistent, we can immediately conclude that there is independent motion involved between the two frames. On the other hand, if the linear system is consistent, we may presume that there is no independent motion. This presumption is subject to a potential false negative, as the linear consistency is not the sufficient condition for the independent motion detection. Similarly, if a large computation noise occurs (e.g., the image point localization errors, the displacement vector estimation errors), a consistent linear system could turn out to be inconsistent. In this case, a false positive would be generated. In general, a few false positives are allowed while the number of false negatives must be guaranteed to a minimum.

Now the question is, given n points in two frames, how to determine whether the linear system Eq. 7 is consistent. By linear algebra theory [25], Eq. 7 is consistent *iff*

$$Rank(\boldsymbol{D}) = Rank(\boldsymbol{Db}) \tag{8}$$

where $\boldsymbol{Db}$ is the augmented matrix of Eq. 7. In order to determine the rank of the above two matrices, we apply singular value decomposition (SVD) to both $\boldsymbol{D}$ and $\boldsymbol{Db}$, and define

$$R = \frac{\sigma_{min}(\boldsymbol{D})}{\sigma_{min}(\boldsymbol{Db})} \tag{9}$$

where $\sigma_{min}(\boldsymbol{D})$ and $\sigma_{min}(\boldsymbol{Db})$ are the smallest singular values of $\boldsymbol{D}$ and $\boldsymbol{Db}$, respectively, assuming Eq. 7 has unique solution; multiple solution cases may be handled similarly. Consequently, Eq. 7 is consistent *iff* R is above a threshold, following the theory and practice of [49, 50, 48]. Note that from the definition of Eq. 9, $R \geq 1$.

There are two comments that are worth mentioning regarding the computation of the linear consistency of Eq. 7.

- The transform Eq. 1 does not assume the calibration of the camera, allowing the use of uncalibrated image coordinates x_i, y_i in the linear system Eq. 7, and empowering this approach for practical applications, for in typical video data the sensor parameters are unknown.
- The coefficients of the linear system Eq. 7 are all linear terms of $x_i, y_i, \dot{x}_i, \dot{y}_i$. This eliminates the popular but notorious problem of high condition numbers in linear

systems typically existing in many image understanding solutions [17] that makes the solutions unstable. This property indicates that the R statistic defined in Eq. 9 is stable, which is confirmed by the experiments.

3.2 LSCA Method

Two potential problems are revealed in the linear system Eq. 7. Eq. 7 assumes the availability of displacement vectors $(\dot{x}, \dot{y})^T$ for a group of image feature points in a frame. In order to obtain the displacement vectors, a correspondence algorithm or an optical flow algorithm must be applied, such as [28, 41, 51]. Since the correspondence or the flow computation problem [2] is an ill-posed problem in computer vision and image understanding research, false positives and false negatives in terms of the correspondence errors between two frames are inevitable, which will be propagated to false positives and false negatives in independent motion detection. Consequently, it is desirable to have a linear system independent of any specific correspondence algorithms or optical flow algorithms for computing the displacement vectors.

The second problem is that the current linear system Eq. 7 assumes the availability of two *static* images, as opposed to two *video frames*. Video frames are typically stored in *compressed data* based on a specific video format such as MPEG, unless they are explicitly *decompressed* to become static images. Therefore, it is also expected to have a linear system directly based on the compressed video data. Below we propose a solution to avoiding the two problems leading to the **LSCA** method based on the MPEG standard[1].

Recall Eqs. 1 and 4. Instead of applying them to a set of feature points in a frame, we now apply them to *every* points of a region of m points in the frame. Thus, we have

$$\sum_{i=1}^{m} \dot{\boldsymbol{p}}_i = \boldsymbol{A}(\boldsymbol{R}-\boldsymbol{I})\sum_{i=1}^{m} \boldsymbol{P}_i + m\boldsymbol{A}\boldsymbol{T} \tag{10}$$

$$\sum_{i=1}^{m} \boldsymbol{p}_i = \boldsymbol{A}\sum_{i=1}^{m} \boldsymbol{P}_i + m\boldsymbol{t} \tag{11}$$

Define

$$\bar{\boldsymbol{p}} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{p}_i = (\bar{x}, \bar{y})^T$$

$$\bar{\dot{\boldsymbol{p}}} = \frac{1}{m}\sum_{i=1}^{m} \dot{\boldsymbol{p}}_i = (\bar{\dot{x}}, \bar{\dot{y}})^T$$

$$\bar{\boldsymbol{P}} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{P}_i = (\bar{X}, \bar{Y}, \bar{Z})^T$$

we obtain

$$\bar{\dot{\boldsymbol{p}}} = \boldsymbol{A}(\boldsymbol{R}-\boldsymbol{I})\bar{\boldsymbol{P}} + \boldsymbol{A}\boldsymbol{T} \tag{12}$$

$$\bar{\boldsymbol{p}} = \boldsymbol{A}\bar{\boldsymbol{P}} + \boldsymbol{t} \tag{13}$$

[1] In this report, we only consider MPEG-1 or MPEG-2. However, the principles apply to other compression standards that might have different structures but the same general motion-vector philosophies.

If we take each MPEG macroblock as such a region, then m becomes a constant (i.e., m = 256) over the whole frame. Therefore, we have a similar linear constraint for *each macroblock* of a frame:

$$\dot{\bar{x}} + \theta\dot{\bar{y}} + \alpha\bar{x} + \beta\bar{y} + \gamma = 0 \tag{14}$$

and consequently, given n macroblocks, we can build a similar linear system

$$\boldsymbol{D_m} = \boldsymbol{\xi_m b_m} \tag{15}$$

with a similar theorem:

Theorem 3.2 *Given n macroblocks in an MPEG video frame represented in the linear system of Eq. 15, if there is no independent motion with any of these macroblocks, then the linear system is consistent.*

In the MPEG compression standard, for each macroblock in a frame, if this macroblock is inter-coded, there is a motion vector available. We approximate $\dot{\bar{\boldsymbol{p}}}$ with the motion vector, and $\bar{\boldsymbol{p}}$ is the center of the macroblock. Since the macroblock information (including the motion vector and the center coordinates) can be easily obtained directly from a compressed MPEG video stream, we have a linear system Eq. 15 that can directly work on the MPEG compressed data without having to depend on a specific algorithm to compute the correspondence or optical flow between the two frames, simultaneously eliminating the two potential problems mentioned above with Eq. 7. If the macroblock is intra-coded, we just exclude this macroblock from the linear system of Eq. 15. If the frame is an I-frame in which all the macroblocks are intra-coded, we can obtain the motion vector of a macroblock by predicting it from the one in the previous B-frame.

While the displacement vector may be approximated by the motion vector of a macroblock, this may create another problem, i.e., how accurate this approximation is. It is known [43] that the motion vector estimation in MPEG is subject to errors, and how large the errors are depends on the specific implementation of the motion vector estimation algorithm under the MPEG standard [8]. The theoretic relationship between the errors in motion vector estimation in MPEG and the detection accuracy is shown in Section 3.3. Here we provide a tentative solution to this problem based on the normal flow computation to attempt to lower the potential errors for the motion estimation. Research shows [47, 39, 27] that the normal flow is more reliable than the standard optical flow. Assuming that the intensity function of a frame is $I(x, y)$, the normal flow n_p at the point $\boldsymbol{p} = (x, y)^T$ is defined as the dot product between the normalized gradient of the point $\boldsymbol{p}$ and the displacement vector at this point:

$$n_p = \frac{\partial I}{\partial x}\dot{x} + \frac{\partial I}{\partial y}\dot{y} \tag{16}$$

Since in the compressed MPEG video stream we only have the motion vectors for each macroblocks as opposed to each points, we must extend this point-based normal flow definition to the macroblock based one. Let $\nabla I(\boldsymbol{p})$ be the normalized gradient of the intensity function I at a point $\boldsymbol{p}$. Given a macroblock $\boldsymbol{M}$, the *macroblock gradient* $\nabla I(\boldsymbol{M})$ is defined as:

$$\nabla I(\boldsymbol{M}) = \frac{1}{m}\sum_{i=1}^{m} \nabla I(\boldsymbol{p_i}) \tag{17}$$

where $\boldsymbol{p}_i$ is a point of $\boldsymbol{M}$, and m is the total number of points in $\boldsymbol{M}$. In MPEG, $m = 256$.

Now the question is how to estimate the gradient of a macroblock without decompressing the video data. Lee et al [26] showed a method of estimating the approximated gradient for a whole block only using a few low frequency AC coefficients of the DCT of the block in MPEG. This is essentially to approximate the original DCT AC coefficients AC_{uv} with the corresponding "continuous" versions $\tilde{AC_{uv}}$:

$$AC_{uv} \approx \tilde{AC_{uv}} = C(u)C(v) \int_0^8 \int_0^8 cos\frac{xu\pi}{8} cos\frac{yv\pi}{8} I(x,y)dxdy \tag{18}$$

where $C(u)$ and $C(v)$ are the scale factors of the standard DCT definition [43]. Given a few limited lower frequency terms of AC_{uv}, we can explicitly solve for the block edge orientation, the block edge offset, and the block edge strength [26]. The question, however, is how many such lower AC coefficients would suffice an accurate estimate of the block gradient. Reported research [26] shows that in order to estimate the block gradient, only the five lowest frequency AC coefficients are necessary to recover the information (i.e., $AC_{01}, AC_{10}, AC_{20}, AC_{11}, AC_{02}$). Consequently, the majority of the AC coefficients as well as the DC component are not required. This shows that it is still not necessary to decompress the video stream in order to recover the block gradient; the method can directly work on the compressed MPEG stream to extract the small piece of the "essential" information (i.e., the motion vector of a macroblock and the five low frequency AC components of a block) without having to decompress the video stream. Fig. 2(a) - (c) shows examples of the "block" edges detected based on estimating the block gradients using the five lowest AC components of the blocks. Note that due to the background noise (e.g., the different background objects in Fig. 2 (a) and (b)), "edge" information is indicated through the block gradient detection; when the background is relatively noise-free (e.g., in Fig. 2 (c)), there is no such "edge" information.

Once we have the block gradient vectors available for all the four blocks of a macroblock, the macroblock gradient is computed by averaging the four block gradient vectors based on the definition. Finally, the normal flow value of a macroblock, $n(\boldsymbol{M})$, is defined similar to that of a point in Eq. 16 by taking the dot product between the macroblock gradient vector, $\nabla I(\boldsymbol{M})$, and the motion vector of this macroblock, $V(\vec{\boldsymbol{M}})$

$$n(\boldsymbol{M}) = \nabla I(\boldsymbol{M}) \cdot V(\vec{\boldsymbol{M}}) \tag{19}$$

When we have the normal flow value computed for a macroblock, we can make a decision regarding whether this macroblock should be incorporated into the linear system of Eq. 15. The rationale [47] is that if the normal flow is low, the motion vector is probably not accurately estimated; consequently this macroblock should be rejected from incorporating into Eq. 15.

Now the **LSCA** algorithm is summarized as follows, which takes four parameters: the normal flow threshold T_n, the scan window width r, the R statistic threshold T_R, and the defined minimum number of frames T_f of a segment that contains independent motion.

```
Scan an input video stream in compressed MPEG
For every pair of neighboring frames
    Start to build up the linear system Eq. 15
    For each macroblock $M$ of the first frame $l$ of the pair
```

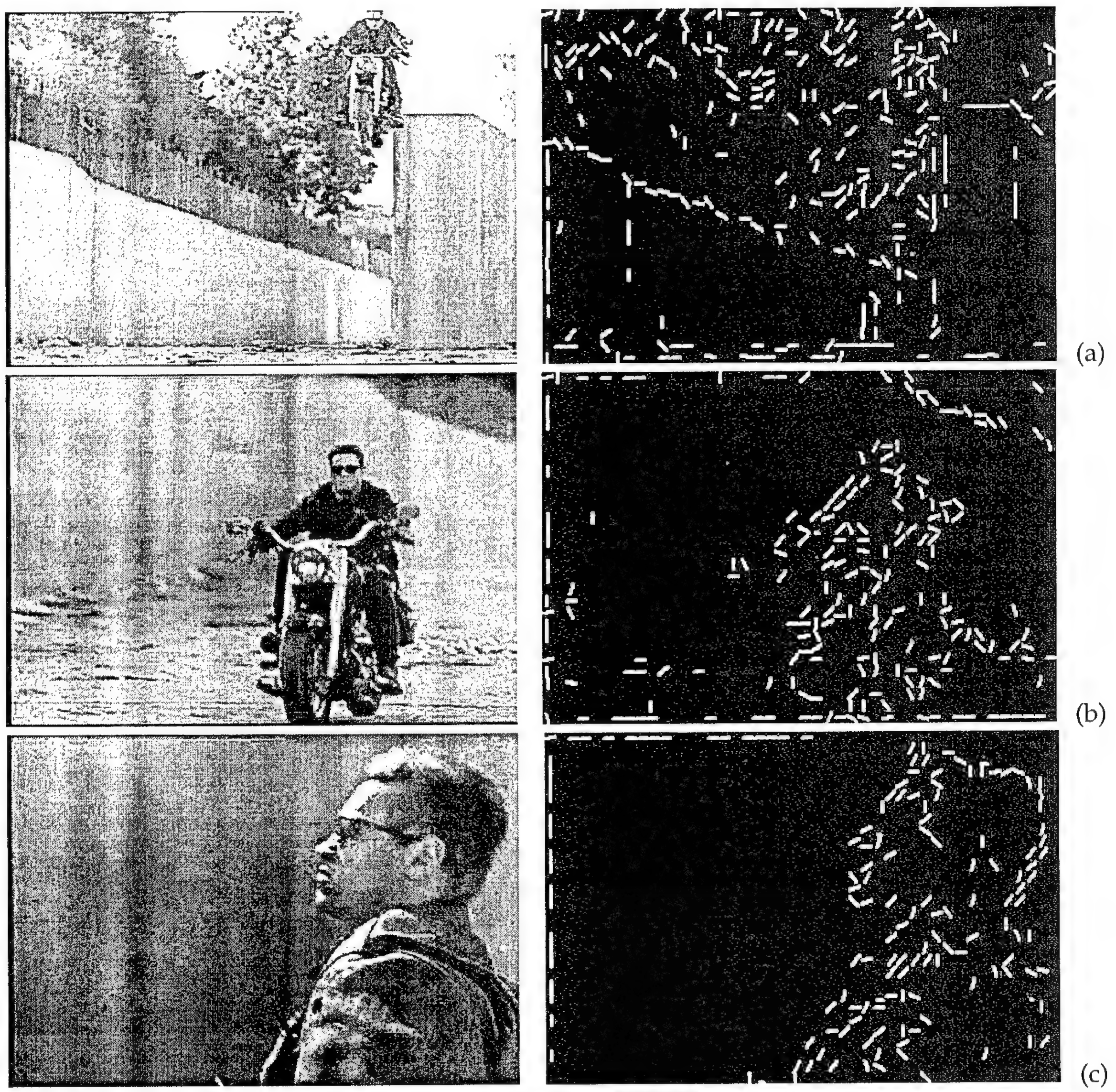


Figure 2: Examples of using the five lowest DCT AC coefficients in each block to estimate the gradient information, where the left column shows the frame samples while the right column shows the corresponding block edge maps. Note that no decompression is performed.

```
        Estimate the normal flow $n(M)$ of $M$
        If $n(M) > T_n$
            Incorporate $M$ into Eq. 15 based on Eq. 14
    Compute $R$ of the linear system Eq. 15
Compute the median filtered $\bar{R}$ over a window of $r$ frames
If $\bar{R} - 1 > T_R$
    Label $l$ as a frame with no independent motion
Else, label $l$ as a frame with independent motion
Any independent motion segment with frame number $> T_f$ is retrieved
```

Note that **LSCA** is based on the assumption of constant camera model in terms of the sensor parameters $\boldsymbol{A}$ and $\boldsymbol{t}$. In real applications, it is possible that the camera internal parameters change during the surveillance (e.g., zoom in/out). Since **LSCA** only focuses on two neighboring frames, given the current video frame rate (about 30 frames/second), if the change is slow, we can ignore the change and still use the algorithm to compute the R statistic between the two frames; if the change is fast, the computed R value between the two frames may be wrong, which will lead to a false positive or negative. However, in this case, there will be only a few frames subject to the error of R values, and they will be shown as outliers and will then typically be filtered out by **LSCA**.

Based on the analysis given above, it is clear that as a novel unsupervised video segmentation tool for independent motion detection, **LSCA** has the following distinctive advantages as compared with the existing methods in the literature:

- No camera calibration is required or necessary in order to apply **LSCA**, i.e., the uncalibrated image coordinates directly from the video frame may be used without having to convert them into the calibrated coordinates.
- The statistics computed in **LSCA** are stable due to the low condition number in the linear system, resulting in avoiding the unstable matrix computation problem of high condition numbers typically existing in many computer vision and image understanding techniques.
- **LSCA** is able to detect independent motion only based on two frames, as opposed to some techniques in the literature requiring more than two frames (e.g., the methods by Irani and Anandan [19] and by Sawhney et al [38]). Those that require more than two frames typically attempt to do spatial identification of the specific independently moving objects in the frames.
- **LSCA** is very fast. The current prototype with even-not-optimized-yet implementation runs at 35 frames/second rate for a typical frame resolution of 240×350 of compressed MPEG videos on a Pentium III 800 MHz Windows2000 system with 512 MB memory.
- **LSCA** directly works on the compressed data without having to decompress the data first.
- **LSCA** only requires one camera video stream for robust detection as opposed to some techniques in the literature requiring the stereo video streams (e.g., the techniques proposed by Argyros et al [5, 3, 4, 6] and by Lourakis et al [27]).

3.3 Theoretic Detection Bound of LSCA Method

In this subsection we give a theoretic bound on independent motion detection using **LSCA** w.r.t. the MPEG motion vector errors. To simplify the notations, in the rest of the report, we will drop the subscript m for the matrices $\boldsymbol{D_m}$, $\boldsymbol{D_m b_m}$, and the vectors $\boldsymbol{\xi_m}$, $\boldsymbol{b_m}$ in Eq. 15, and will drop the overline for the coordinates and the motion vector components for the center of a macroblock, i.e., $\bar{\boldsymbol{p}} = (\bar{x}, \bar{y})^T \Longrightarrow \boldsymbol{p} = (x, y)^T$, and $\dot{\bar{\boldsymbol{p}}} = (\dot{\bar{x}}, \dot{\bar{y}})^T \Longrightarrow \dot{\boldsymbol{p}} = (\dot{x}, \dot{y})^T$.

Define

$$\tilde{\dot{\boldsymbol{p}}} = \dot{\boldsymbol{p}} + \Delta\dot{\boldsymbol{p}} = (\dot{x}, \dot{y})^T + (\Delta\dot{x}, \Delta\dot{y})^T$$

as the motion vector given in the MPEG streams, which is assumed here to be decomposed into the true motion vector $\dot{\boldsymbol{p}} = (\dot{x}, \dot{y})^T$ and the error $\Delta\dot{\boldsymbol{p}} = (\Delta\dot{x}, \Delta\dot{y})^T$. Since we assume that the motion vector estimation error in the MPEG streams is the only error source contributing to the detection error in the **LSCA** approach, we have

$$(\boldsymbol{D} + \triangle\boldsymbol{D})\boldsymbol{\xi} = \boldsymbol{b} + \triangle\boldsymbol{b} \tag{20}$$

where

$$\boldsymbol{D} = \begin{pmatrix} \dot{y_1} & x_1 & y_1 & 1 \\ \dots & & & \dots \\ \dot{y_n} & x_n & y_n & 1 \end{pmatrix} \quad \triangle\boldsymbol{D} = \begin{pmatrix} \Delta\dot{y_1} & 0 & 0 & 0 \\ \dots & & & \dots \\ \Delta\dot{y_n} & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} -\dot{x_1} \\ \dots \\ -\dot{x_n} \end{pmatrix} \quad \triangle\boldsymbol{b} = \begin{pmatrix} -\Delta\dot{x_1} \\ \dots \\ -\Delta\dot{x_n} \end{pmatrix}$$

Define the augmented matrix

$$\boldsymbol{H} = \boldsymbol{Db} = \begin{pmatrix} \dot{y_1} & x_1 & y_1 & 1 & -\dot{x_1} \\ \dots & & & & \dots \\ \dot{y_n} & x_n & y_n & 1 & -\dot{x_n} \end{pmatrix}$$

and further define

$$\boldsymbol{H'} = \boldsymbol{H} + \triangle\boldsymbol{H} = (\boldsymbol{D} + \triangle\boldsymbol{D})|(\boldsymbol{b} + \triangle\boldsymbol{b})$$

resulting in

$$\triangle\boldsymbol{H} = \begin{pmatrix} \Delta\dot{y_1} & 0 & 0 & 0 & -\Delta\dot{x_1} \\ \dots & & & & \dots \\ \Delta\dot{y_n} & 0 & 0 & 0 & -\Delta\dot{x_n} \end{pmatrix}$$

We introduce the following notations. For a matrix $\boldsymbol{B}$, we denote $\lambda_i(\boldsymbol{B})$ as the ith eigenvalue of the matrix $\boldsymbol{B}$, and $\sigma_i(\boldsymbol{B})$ as the ith singular value of the matrix $\boldsymbol{B}$. In particular, we denote $\lambda_{min}(\boldsymbol{B})$ as the smallest eigenvalue of the matrix $\boldsymbol{B}$, and $\sigma_{min}(\boldsymbol{B})$ as the smallest singular value of the matrix $\boldsymbol{B}$. We assume that for a matrix $\boldsymbol{B}$, all the eigenvalues or the singular values are sorted from the largest to the smallest.

From the perturbation theory of singular value decomposition [16], we have

$$\|\sigma_{min}(\boldsymbol{H} + \triangle\boldsymbol{H}) - \sigma_{min}(\boldsymbol{H})\| \leq \sigma_1(\triangle\boldsymbol{H}) \tag{21}$$

$$\|\sigma_{min}(\boldsymbol{D} + \triangle\boldsymbol{D}) - \sigma_{min}(\boldsymbol{D})\| \leq \sigma_1(\triangle\boldsymbol{D}) \tag{22}$$

From the relationship between the eigenvalues and the singular values of the corresponding matrices [16], we have

$$\sigma_1^2(\triangle\boldsymbol{H}) = \lambda_1(\triangle\boldsymbol{H}^T\triangle\boldsymbol{H}) \tag{23}$$

$$\sigma_1^2(\triangle \boldsymbol{D}) = \lambda_1(\triangle \boldsymbol{D}^T \triangle \boldsymbol{D}) \tag{24}$$

By explicitly solving for the eigenvalues of the matrices $\triangle \boldsymbol{H}^T \triangle \boldsymbol{H}$ and $\triangle \boldsymbol{D}^T \triangle \boldsymbol{D}$, we have

$$\sigma_1(\triangle \boldsymbol{H}) = \sqrt{\frac{\sum_{i=1}^n \triangle \dot{y}_i^2 + \sum_{i=1}^n \triangle \dot{x}_i^2 + \sqrt{(\sum_{i=1}^n \triangle \dot{y}_i^2 - \sum_{i=1}^n \triangle \dot{x}_i^2)^2 + 4(\sum_{i=1}^n \triangle \dot{x}_i \triangle \dot{y}_i)^2}}{2}} \tag{25}$$

$$\sigma_1(\triangle \boldsymbol{D}) = \sqrt{\sum_{i=1}^n \triangle \dot{y}_i^2} \tag{26}$$

Based on the definition in Eq. 9, now we have

$$\begin{aligned} \triangle R &= \sum_{i=1}^n \frac{\partial R}{\partial \dot{x}_i} \triangle \dot{x}_i + \sum_{i=1}^n \frac{\partial R}{\partial \dot{y}_i} \triangle \dot{y}_i \\ &= \sum_{i=1}^n \frac{\frac{\partial \sigma_{min}(\boldsymbol{D})}{\partial \dot{x}_i} \sigma_{min}(\boldsymbol{H}) - \frac{\partial \sigma_{min}(\boldsymbol{H})}{\partial \dot{x}_i} \sigma_{min}(\boldsymbol{D})}{\sigma_{min}^2(\boldsymbol{H})} \triangle \dot{x}_i \\ &+ \sum_{i=1}^n \frac{\frac{\partial \sigma_{min}(\boldsymbol{D})}{\partial \dot{y}_i} \sigma_{min}(\boldsymbol{H}) - \frac{\partial \sigma_{min}(\boldsymbol{H})}{\partial \dot{y}_i} \sigma_{min}(\boldsymbol{D})}{\sigma_{min}^2(\boldsymbol{H})} \triangle \dot{y}_i \\ &= \frac{1}{\sigma_{min}(\boldsymbol{H})} \sum_{i=1}^n \left(\frac{\partial \sigma_{min}(\boldsymbol{D})}{\partial \dot{x}_i} \triangle \dot{x}_i + \frac{\partial \sigma_{min}(\boldsymbol{D})}{\partial \dot{y}_i} \triangle \dot{y}_i\right) \\ &- \frac{\sigma_{min}(\boldsymbol{D})}{\sigma_{min}^2(\boldsymbol{H})} \sum_{i=1}^n \left(\frac{\partial \sigma_{min}(\boldsymbol{H})}{\partial \dot{x}_i} \triangle \dot{x}_i + \frac{\partial \sigma_{min}(\boldsymbol{H})}{\partial \dot{y}_i} \triangle \dot{y}_i\right) \end{aligned} \tag{27}$$

Based on the calculus theory, let

$$\| \sum_{i=1}^n \left(\frac{\partial \sigma_{min}(\boldsymbol{D})}{\partial \dot{x}_i} \triangle \dot{x}_i + \frac{\partial \sigma_{min}(\boldsymbol{D})}{\partial \dot{y}_i} \triangle \dot{y}_i\right) \| \simeq \| \sigma_{min}(\boldsymbol{D} + \triangle \boldsymbol{D}) - \sigma_{min}(\boldsymbol{D}) \| \tag{28}$$

and

$$\| \sum_{i=1}^n \left(\frac{\partial \sigma_{min}(\boldsymbol{H})}{\partial \dot{x}_i} \triangle \dot{x}_i + \frac{\partial \sigma_{min}(\boldsymbol{H})}{\partial \dot{y}_i} \triangle \dot{y}_i\right) \| \simeq \| \sigma_{min}(\boldsymbol{H} + \triangle \boldsymbol{H}) - \sigma_{min}(\boldsymbol{H}) \| \tag{29}$$

Consequently, from Eq. 27, we have

$$\begin{aligned} \| \triangle R \| &\leq \| \frac{1}{\sigma_{min}(\boldsymbol{H})} \sum_{i=1}^n \left(\frac{\partial \sigma_{min}(\boldsymbol{D})}{\partial \dot{x}_i} \triangle \dot{x}_i + \frac{\partial \sigma_{min}(\boldsymbol{D})}{\partial \dot{y}_i} \triangle \dot{y}_i\right) \| \\ &+ \| \frac{\sigma_{min}(\boldsymbol{D})}{\sigma_{min}^2(\boldsymbol{H})} \sum_{i=1}^n \left(\frac{\partial \sigma_{min}(\boldsymbol{H})}{\partial \dot{x}_i} \triangle \dot{x}_i + \frac{\partial \sigma_{min}(\boldsymbol{H})}{\partial \dot{y}_i} \triangle \dot{y}_i\right) \| \end{aligned} \tag{30}$$

Since $\sigma_{min}(\boldsymbol{D}) > 0, \sigma_{min}(\boldsymbol{H}) > 0$, from Eqs. 21, 22, 25, 26, 28, and 29, we have the theoretic error bound for R:

$$\begin{aligned} \| \triangle R \| \leq & \frac{\sqrt{\sum_{i=1}^n \triangle \dot{y}_i^2}}{\sigma_{min}(\boldsymbol{H})} \\ & + \frac{\sigma_{min}(\boldsymbol{D})}{\sigma_{min}^2(\boldsymbol{H})} \sqrt{\frac{\sum_{i=1}^n \triangle \dot{y}_i^2 + \sum_{i=1}^n \triangle \dot{x}_i^2 + \sqrt{(\sum_{i=1}^n \triangle \dot{y}_i^2 - \sum_{i=1}^n \triangle \dot{x}_i^2)^2 + 4(\sum_{i=1}^n \triangle \dot{x}_i \triangle \dot{y}_i)^2}}{2}} \end{aligned} \tag{31}$$

Experimental data have shown that this error bound is consistent very well with the error distributions in the data.

4 Frame Alignment

Frame alignment has many immediate applications, including video mosaicking, frame stabilization, spatial domain object tracking, etc. Since frame alignment is another typical video prostanding problem, we show that this problem may also be solved using the holistic, in-compression approach as a tool for video summarization (e.g., through the mosaicking application to have immediate access to the "background" scene without playing the video). The specific theory and the related method for frame alignment developed here is called In-Compression Matching (**ICM**). Below we first discuss the theory, and then introduce the **ICM** method.

As with **LSCA**, here we will make use of the motion vectors in the MPEG-compressed video. In a later subsection we discuss the details of dealing with I-, P-, and B-frames in the sequence. For the moment, however, we will suppose that for the current frame, there are n motion vectors received; note n may vary from frame to frame due to an encoder's selecting macroblocks to be coded in intra mode; these motion vectors will be used to align the current frame to the reference frame. Let the i^{th} motion vector be $\mathbf{v_i} = (\mathbf{\dot{x}_i}, \mathbf{\dot{y}_i})^\mathbf{T}$. It is assumed here that $\mathbf{v_i}$ describes the motion undergone by the pixel located at the center of the macroblock, denoted as $(x_i, y_i)^T$, such that the current macroblock is centered at position $(\hat{x}_i, \hat{y}_i)^T = (x_i + \dot{x}_i, y_i + \dot{y}_i)^T$ in the reference frame.

We use an affine transformation in two dimensions to represent the global motion between two frames,

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \tag{32}$$

which contains six degrees of freedom. The four- and two-parameter models discussed in this report have the same general form as the affine model above, but with varying degrees of freedom removed. A more general eight-parameter projective transform could also be considered, but the model's non-linear solution is avoided in favor of the more efficient performance of the models discussed in this section. Efficient implementation of estimation of the eight-parameter projective model [29, 36] has been addressed elsewhere, but is not investigated here.

4.1 Six-Parameter Model

A weighted least-squares (LS) formulation is first introduced for fitting the received MPEG motion vectors to the affine transform. Inclusion of the weights in the formulation allows an implementation to include additional information about the accuracy of some motion vectors relative to others. Appropriate weighting of the motion vectors can help to prevent inaccuracies of the motion vectors from corrupting the estimated motion.

The weighted LS formulation for the affine model minimizes

$$\sum_{i=1}^{n} w_i(a_{11}x_i + a_{12}y_i + a_{13} - \hat{x}_i)^2 + \sum_{i=1}^{n} w_i(a_{21}x_i + a_{22}y_i + a_{23} - \hat{y}_i)^2, \tag{33}$$

which is accomplished by solving the following two equations:[2]

$$\begin{bmatrix} \sum w_i x_i^2 & \sum w_i x_i y_i & \sum w_i x_i \\ \sum w_i x_i y_i & \sum w_i y_i^2 & \sum w_i y_i \\ \sum w_i x_i & \sum w_i y_i & \sum w_i \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} = \begin{bmatrix} \sum w_i \hat{x}_i x_i \\ \sum w_i \hat{x}_i y_i \\ \sum w_i \hat{x}_i \end{bmatrix}, \tag{34}$$

$$\begin{bmatrix} \sum w_i x_i^2 & \sum w_i x_i y_i & \sum w_i x_i \\ \sum w_i x_i y_i & \sum w_i y_i^2 & \sum w_i y_i \\ \sum w_i x_i & \sum w_i y_i & \sum w_i \end{bmatrix} \begin{bmatrix} a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} \sum w_i \hat{y}_i x_i \\ \sum w_i \hat{y}_i y_i \\ \sum w_i \hat{y}_i \end{bmatrix}. \tag{35}$$

The square matrix on the left of the above two equations is easily inverted, yielding the weighted least-squares estimate of the six affine parameters.

Extreme outliers in the MPEG motion vectors can corrupt the LS model fitting. A considerably more robust optimization minimizes the weighted least absolute value (LAV) error criterion,

$$\sum w_i |a_{11} x_i + a_{12} y_i + a_{13} - \hat{x}_i| + \sum w_i |a_{21} x_i + a_{22} y_i + a_{23} - \hat{y}_i|. \tag{36}$$

Such an error criterion is well-known to be more robust to the effects of outliers in one's data. Here, the LAV estimate is computed using iteratively reweighted least squares (IRLS), which makes use of the weighted LS solution shown previously. Since the two terms in Eq. 36 can be minimized independently, we describe the procedure for minimization of the x error term; minimization of the other term in performed analogously. The IRLS solution starts with an initial estimate of the affine parameters, taken here as the standard weighted least-squares estimate. The absolute value of the residual error is then computed, and a new weight is defined as

$$\hat{w}_i^{(j)} = \frac{w_i}{\left|a_{11}^{(j)} x_i + a_{12}^{(j)} y_i + a_{13}^{(j)} - \hat{x}_i\right|}, \tag{37}$$

where the superscript (j) denotes values for the j^{th} iteration of the algorithm. This weight is then used in place of w_i to compute a new weighted least squares estimate. This process repeats iteratively until a convergence criterion is met, taken here as

$$\frac{\sum |r_i^{(j)}| - \sum |r_i^{(j-1)}|}{\sum |r_i^{(j-1)}|} < \epsilon, \tag{38}$$

where $r_i^{(j)}$ is the residual error term in the denominator of Eq. 37, and ϵ is a threshold. Convergence has been observed to occur in two to six iterations.

4.2 Four-Parameter Model

It is well-known that using a model with more degrees of freedom than is strictly necessary can lead to the model simply fitting the *noise* more accurately. In some scenarios, the six parameters of the affine transform may be more than is necessary to describe the motion accurately, and the additional robustness of a reduced-parameter model to motion vector inaccuracies can make a lower-order model a better choice than the affine model. This subsection considers the case of a four-parameter model.

[2]For notational convenience the indices for the summation have been removed, and are implicitly meant to be over i from 1 to n.

When one applies rotation, followed by zoom, followed by translation, the resulting transformation appears as

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix} = \begin{bmatrix} a & -b & d_x \\ b & a & d_y \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \tag{39}$$

where the parameters can be interpreted such that $\alpha = \sqrt{a^2 + b^2}$ is the zoom, $\theta = \tan^{-1} \frac{b}{a}$ is the angle of rotation, and d_x and d_y are the translational shifts in the x and y directions. The weighted LS solution for the four-parameter model minimizes

$$\sum w_i \left[(ax_i - by_i + d_x - \hat{x}_i)^2 + (bx_i + ay_i + d_y - \hat{y}_i)^2 \right], \tag{40}$$

whose minimum is found by solving

$$\begin{bmatrix} \sum w_i(x_i^2 + y_i^2) & 0 & \sum w_i x_x & \sum w_i y_i \\ 0 & \sum w_i(x_i^2 + y_i^2) & -\sum w_i y_i & \sum w_i x_i \\ \sum w_i x_i & -\sum w_i y_i & \sum w_i & 0 \\ \sum w_i y_i & \sum w_i x_i & 0 & \sum w_i \end{bmatrix} \begin{bmatrix} a \\ b \\ d_x \\ d_y \end{bmatrix} = \begin{bmatrix} \sum w_i(x_i \hat{x}_i + y_i \hat{y}_i) \\ \sum w_i(-y_i \hat{x}_i + x_i \hat{y}_i) \\ \sum w_i \hat{x}_i \\ \sum w_i \hat{y}_i \end{bmatrix}. \tag{41}$$

The inverse of the above matrix is easily computed as

$$\frac{1}{-z_{11}z_{33} + z_{13}^2 + z_{14}^2} \begin{bmatrix} -z_{33} & 0 & z_{13} & z_{14} \\ 0 & -z_{33} & -z_{14} & z_{13} \\ z_{13} & -z_{14} & -z_{11} & 0 \\ z_{14} & z_{13} & 0 & -z_{11} \end{bmatrix}, \tag{42}$$

where z_{ij} corresponds to the $(i,j)^{th}$ element of the square matrix in Eq. 41.

As was the case for the affine model, a LAV estimate can be computed for the four-parameter model. The procedure is the same as for the affine model, but the weighting term from Eq. 37 is replaced here by

$$\hat{w}_i^{(j)} = w_i \frac{|r_{x_i}^{(j)}| + |r_{y_i}^{(j)}|}{\left[r_{x_i}^{(j)}\right]^2 + \left[r_{y_i}^{(j)}\right]^2}, \tag{43}$$

where $r_{x_i}^{(j)}$ and $r_{y_i}^{(j)}$ are the i^{th} x and y residual error terms in Eq. 40 for the j^{th} iteration.

4.3 Two-Parameter Model

Just as the four-parameter model may often be more appropriate than the six-parameter model, at times the two-parameter model may be more appropriate than the higher-order models. The two-parameter model of this subsection is extremely robust to inaccuracies in MPEG motion vectors, and is the preferred method when the global motion is well-approximated by simple translational shifts.

The two parameter model includes only the two translational parameters d_x and d_y,

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}. \tag{44}$$

Use of the two-parameter model is appropriate when the global motion is approximately translational, for example when one has aerial motion imagery taken with a lookpoint straight down. Some methods have used a two-parameter motion model as an approximation to camera pan and tilt, and under some circumstances when it may not be strictly appropriate, the two-parameter model may still be sufficiently accurate for some particular applications. The two-parameter model is also more robust to errors in observed MPEG motion vectors.

Following the formulation of the four- and six-parameter cases above, one could easily derive a weighted LS solution for the two-parameter case. However, direct application of the LAV estimator is straightforward for the two-parameter case. Minimizing the weighted LAV criterion

$$\sum w_i|x_i + d_x - \hat{x}_i| + \sum w_i|y_i + d_y - \hat{y}_i|, \tag{45}$$

leads to estimates for d_x and d_y being the weighted medians of the x and y components of the received motion vectors.

Note that the weighted median filter can be implemented very efficiently here because sorting the motion vectors is unnecessary—since the MPEG motion vectors are given at half-pixel precision within a range of limited size [20], the cumulative distribution function (cdf) of the weighted motion vectors can be easily computed, with the median value taken as the point of the cdf at which half the weighted data are above that point and half the weighted data are below. Calculating the median of these potentially large numbers of motion vectors can thus be done in $O(n)$ complexity.

4.4 ICM Method

Although inclusion of all frames for frame alignments from an MPEG stream would be straightforward, the application described here (mosaics) primarily uses P-frames; making use of motion vectors from B-frames can be accomplished analogously to the case of P-frames. Motion vectors for each P-frame are used to estimate parameters of a global motion model between the P-frame and its reference frame; these global motions are calculated throughout the sequence. When using only P-frames there is a temporal discontinuity in the motion estimates when an I-frame occurs. This discontinuity is easily remedied if there is at least one B-frame at the end of the previous GOP that uses both forward and backward prediction—the global motion from the B-frame to the P-frame can be computed and used in conjunction with the global motion from the B-frame to the I-frame to form an estimate of the motion from the I-frame to the previous P-frame. If such a B-frame is unavailable, we have one of two options: we can do some form of interpolation of the available motion fields from either temporal side of the I-frame, or we can default to a spatial-domain frame registration algorithm.

Since in MPEG there is a motion vector available for each macroblock if this macroblock is inter-coded, we can directly make use of this motion vector as an approximation to the displacement vector for this macroblock. If the macroblock is intra-coded, there are two cases. We have already discussed the strategy to handle the case when the macroblock comes from an I-frame. If the macroblock is not in an I-frame, we treat the weight for the displacement vector of this macroblock as 0, and thus the macroblock is excluded in the consideration of the frame alignment.

In Section 3.2 we have shown that we can directly obtain the block gradient information from the compressed domain. We can take the same approach to estimating the macroblock gradient information in determining the weights in generating the alignment in the **ICM** method based on the theory discussed above: macroblocks that have more edge activity, as indicated by the gradient, can be weighted more heavily in the formulation than macroblocks with low edge activity whose motion information is less likely to be accurate.

4.5 Example Use of ICM: Mosaics

As mentioned previously frame alignment has many potential uses, but here we focus on the formation of mosaics. The algorithm presented here begins by placing a user-specified reference frame on the blank mosaic canvas, and then proceeds by adding strips of new information provided by other frames in the sequence. For time- and memory-critical applications, the first frame of the sequence can be considered the reference frame, with new information from subsequent frames being added over the frames in the stream. For the more general case, an arbitrary frame can be used as the reference, with the alignment formed by adding the new information provided by frames both before and after the reference frame. For many sequences, various choices of reference frames can have dramatic effects on the final mosaic; some examples are given in Section 5.

4.6 Using the I-Frames Only

The above **ICM** and mosaicking theory assumes that we use both the I- and P-frames in a video sequence. In fact, if we intend to save further time in computation, we can apply the same **ICM** theory to generating a mosaic using only the I-frames. The idea is that we estimate the global motions for each I-frame through the composition of all the global motions from the previous P-frames. This strategy significantly saves the computation time for generating a mosaic. Experimental data show that if the original video sequence motion vectors are of reasonably good quality, the quality of the generated frame alignment is still acceptable. This is particularly attractive for the application scenarios that motivate this project, in which we only need to let the users have a quick browse about the content of the video without having to have a high accuracy of the alignment.

5 Experimental Evaluations

In this section, we first report a preliminary, simulation based analysis on estimating the detection false positives and false negatives, as well as the detectability bounds for **LSCA**. We then present the real data experimental evaluations to demonstrate the robustness and effectiveness of **LSCA**. Finally, we present the real data evaluations for **ICM**.

5.1 Simulation Analysis on LSCA Method

Both analyses on false positives and false negatives are the *sensitivity* analysis for **LSCA**. This may be achieved by testing the *stability* of the statistic R of **LSCA** under different levels of noise through simulation. As a preliminary analysis, we design a simulation scenario as follows.

10 3D points are randomly generated in a Euclidean world coordinate system. A video camera is first located at an arbitrary position in this world coordinate system. The 10 points are projected to the image plane of the camera based on the 3D to 2D affine transform to generate the image coordinates in this frame. Then the camera is moved to another position, and the 10 3D points are projected to the image plane again to generate the image coordinates in the second frame. The image displacement vectors are immediately obtained from the image coordinates of the two frames. The displacement vectors and the image coordinates in the first frame are then corrupted by different levels of randomly generated Gaussian noise, parameterized by different deviations in terms of the number of pixels. The image points in the two frames are distributed in an area of 100 pixels by 120 pixels. Thus, 1 pixel deviation of Gaussian noise approximately corresponds to 1% of the *whole* effective image dimension.

The corrupted image coordinates and the displacement vectors are input into **LSCA**, and the R statistic is computed for each noise level. Since there is no independent motion involved in this scenario, the R value should be high. Under the corruption of the noise, however, the R value degrades as the noise level increases. Fig. 3(a) shows the logarithm of the R values averaged over 1000 runs with different seeds under each Gaussian noise level parameterized by the standard deviation in terms of the number of pixels.

From Fig. 3(a), it is clear that false positives of independent motion detection may occur when the noise level increases. From the Figure, if the noise level is controlled under 2 pixels, the R value always stabilizes somewhere statistically significantly higher than 1 (above 2). Note that considering the effective image dimension as 100 by 120, 2 pixels' noise is significantly large in practice. This shows that **LSCA** is very robust in rejecting false positives in independent motion detection.

The simulation scenario continues when a point of independent motion is added into the original 10 point set. This time the R value always stays at 1 regardless of what level the noise is, indicating **LSCA** is effective in detecting independent motion.

While false positives and false negatives are the probabilities describing an event of a detection failure of the **LSCA**, detectability is an issue of how significant an independent motion should be such that **LSCA** is able to detect it. *Detectability* is a different but a related concept, which is defined as the smallest independent motion that **LSCA** can detect. Again, this may be determined through simulation analysis.

Using the same simulation scenario above, based on the original 10 background points, we add the 11th point that is subject to an independent motion, *in addition to* the camera motion. In order to separate contributions from different independent motion components to the detectability of **LSCA**, we apply **LSCA** to independent translations of this point along X, Y, Z axes, and to independent rotations of this point about X, Y, Z axes, respectively, under different levels of Gaussian noise. Fig. 3(b) to (g) show the six scenarios of the simulation.

A qualitative examination of these simulation results reveals that the performance of **LSCA** appears less sensitive to the independent motion related to the Z axis (rotation about the axis or translation along the axis) than to the independent motion related to the other axes; quantitatively, based on this simulation, the detectability is related to the noise levels, and a higher noise level increases detectability. The reason is that a higher level of noise increases the false positives, which help increase detectability. Take the noise level of 1.5 pixel deviation for example. If the threshold value is set as 2 for R, the detectability is under 1 unit for all the translations, and under 0.1 degree for all the rotations. If the threshold of R decreases to 1.5, the detectability for translations along X, Y, and Z axes is above 8, 10, and 20 units, respectively, for rotations about X, Y, and Z axes is above 0.4, 0.8, and 3.0 degrees, respectively. Note that these parameters are obtained from this specific set of simulation only. However, since in the simulation data we know the ground truth of the displacement vectors, we observe that the R statistic errors propagated from the displacement vector errors confirm well with the theoretic bound in Eq. 31.

5.2 Real Data Evaluations on LSCA Method

We have implemented the **LSCA** as a stand alone version in a Windows2000 platform with Pentium III 800 MHz CPU and 512 MB memory. Fig. 4(a) and (b) show two surveillance video clips for the two scenarios with and without independent motion, and Fig. 4(c) and (d) show the R statistics computed at every frames for the two shots from two surveillance videos in Fig. 4(a) and (b), respectively. The statistics are obvious to tell whether and where there is independent motion in the video. The first shot containing 264 frames describes an independent motion of an airplane landing to its destination. The mean of the R statistics is 1.012 and the deviation is 0.0083 over the 264 frames. The second shot containing 1024 frames surveys an area of ground terrain with no independent motion. The mean of the R statistics is 1.389 and the deviation is 0.169 over the 1024 frames.

In order to give a meaningful evaluation, we make an assumption that a reliable independent motion shot should last at least 30 frames (i.e., $T_f = 30$), which corresponds at least about one second presence of independent motion in the video. This assumption ensures that any sporadic detection false positives due to motion estimation outliers and/or sensor parameter changes will be removed. Since **LSCA** performs frame-based independent motion detection, it is reasonable to define the *detection rate* as the percentage of the number of truthed independent motion frames detected by **LSCA** of the total number of detected independent motion frames, and to define the *detection false alarm* as the percentage of the number of falsely detected independent motion frames reported by **LSCA** of the total number of truthed independent motion frames in a video. Based on these definitions, we have run **LSCA** on a video testbed which collects different shots of surveillance video of total 10602 frames. The overall detection rate is 94.9% and the false alarm is 3.07%.

The above systematic evaluation is based on a threshold value of the R statistic in **LSCA**, which is 1.2. Let us use this threshold as the baseline threshold, and call it T_0. In order to investigate the **LSCA** performance with respect to the varying threshold, we define a variable r_T as the *relative increase*:

$$r_T = \frac{T_R - T_0}{T_0 - 1} \tag{46}$$

where T_R is the varying threshold for the R statistic. We take one of the shots in the

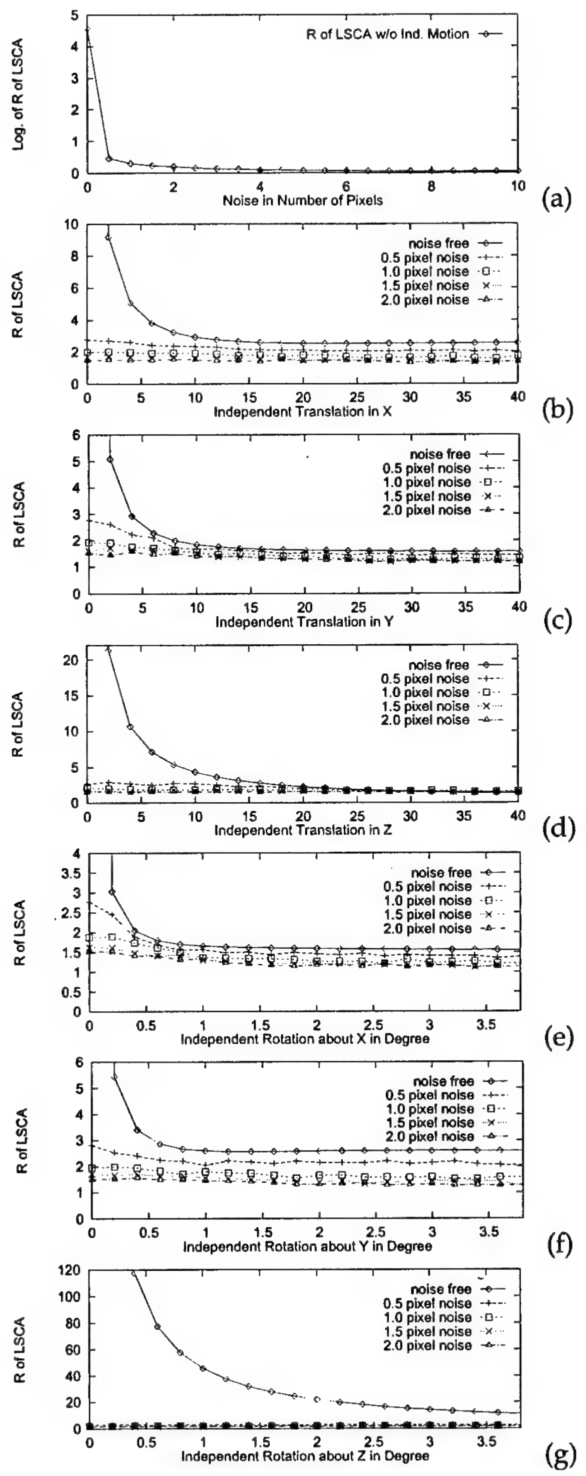
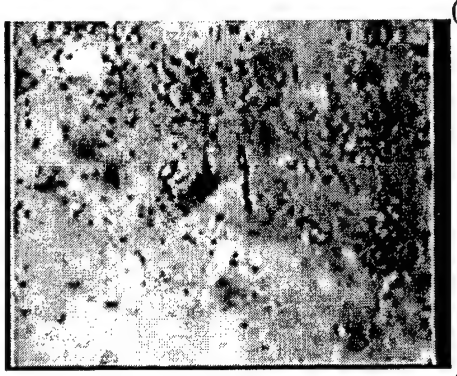


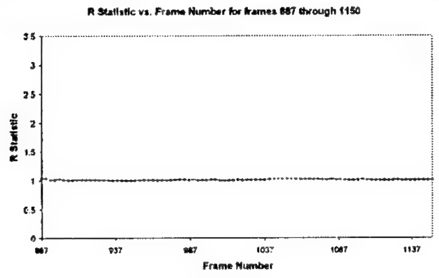
Figure 3: (a) Logarithm of the R statistic of **LSCA** under different Gaussian noise levels when no independent motion involved. (b)-(d) Detectabilities of **LSCA** w.r.t. independent translations along X, Y, and Z axes, respectively. (e)-(g) Detectabilities of **LSCA** w.r.t. independent rotations about X, Y, and Z axes, respectively.

(a)

(b)



(c)



(d)

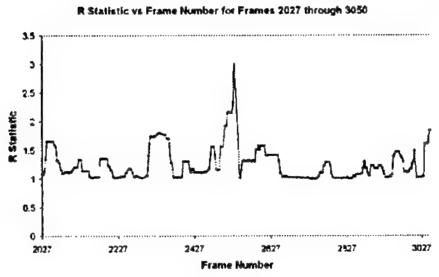


Figure 4: (a) An example of a shot containing an independently moving object (an airplane) (b) An example of a shot containing no independent motion (terrain) (c) The R statistics computed for the shot in (a) (264 frames) (d) The R statistics computed for the shot in (b) (1024 frames).

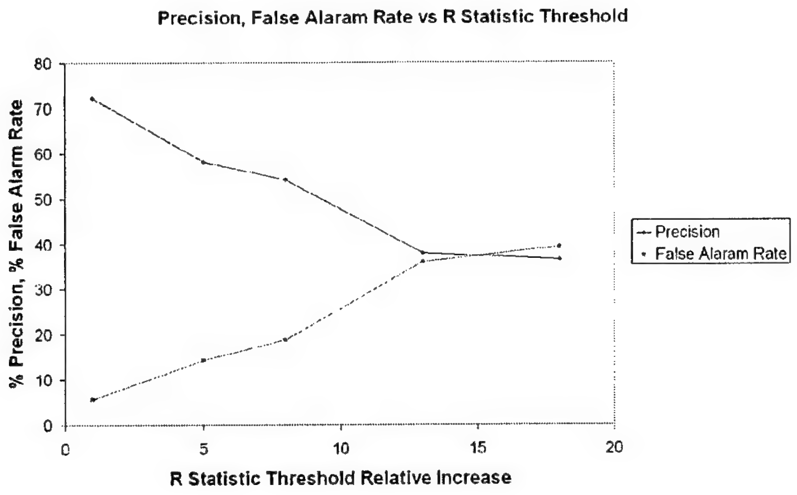


Figure 5: **LSCA** performance degrades gracefully when the r_T increases.

whole evaluation video data collection and apply different relative increases w.r.t. the baseline threshold T_0 to observe the performance degradation. Fig. 5 shows that the **LSCA** performance degrades gracefully when the relative increase r_T increases substantially.

To show that **LSCA** is not only valid for the typical surveillance scenario where the camera is far away from the scene, but also could be valid for the scenario where the camera is relatively close to the scene, Fig. 6 demonstrates an experimental result of **LSCA** in which we take a movie with an independent motion very close to the camera, and split the spatial domain into the left and the right halves such that the left video does not contain independent motion while the right one does. The result clearly shows that **LSCA** is robust even under the situation where the camera is close to the scene.

Since **LSCA** essentially just needs to compute the R value for each frame, and since in each frame there is typically a very limited number of macroblocks, the complexity of **LSCA** is very low. The current prototype of **LSCA** scans a compressed MPEG video with a typical frame resolution of 240 by 350 at the speed of 35 frames/second under the current platform, which is faster than real-time. Note that this implementation is just for proof of the concept and the code has not been optimized yet. This shows that **LSCA** holds a great promise and vitality in the future applications in both proposed scenarios: real time surveillance data scanning equipped with the sensors and efficient data mining for an archived database of surveillance video.

5.3 Real Data Evaluation on ICM Method

The **ICM** method is implemented in a platform of Pentium IV 1.6GHz CPU with 512MB memory running Windows XP. Fig. 7(a) and Fig. 8(a) show examples of the mosaics for two shots with 128 frames each using **ICM** frame alignment method. In order to demonstrate the strength of the **ICM** method, we have also implemented a method that uses the same frame alignment models as **ICM** uses but actually decodes all the frames from the video for the model-fitting. In other words, this method decodes the MPEG stream data for generating the frame alignment instead of directly generating the frame alignment in the compressed domain. After the decoding, the same motion vectors available in the data is used for the model-fitting, and the Sobel edge operator is used to obtain the gradient information for each pixel in the model-fitting in this method. For the reference purpose, we call this method as **DM**. Fig. 7(b) and Fig. 8(b) show the mosaicking results generated using **DM**. Visual examination between Fig. 7 and Fig. 8 indicates that there is little noticeable difference between **ICM** and **DM** regarding the frame alignment quality generated by the two methods. This then demonstrates that weighting using compressed-domain information gives comparable results to using weights derived from actual spatial-domain information.

On the other hand, using **ICM** can substantially save time due to the holistic, in-compression approach. For example, Fig. 7(a) is generated using **ICM** for a video shot of 384 frames with 352×240 frame resolution, and it takes 14 seconds using the six-parameter model, which delivers about 27 frames per second that is close to real-time at the current platform with the not-optimized-yet code. On the other hand, Fig. 7(b) is generated using **DM** for the same video shot, and it takes 30 seconds using the same six-parameter model. This shows the substantial saving in time for the **ICM** method, due to not having to fully decode the frames for alignment.

The accuracy of the generated frame alignments using **ICM** depends on the quality of the motion vectors available in the original video data. This is obvious since the accuracy of the alignment models directly depends on the accuracy of the displacement vectors which are approximated by the motion vectors in **ICM**; for the subsequent frames' alignment parameters, they are generated in **ICM** through compositions through the P-frames from a previous frame's alignments; if there is an error in the previous frame's global motion, this error is propagated and accumulated to larger errors for the global motion in the subsequent frames, resulting in worse accuracy for the alignment quality as the alignment extends to subsequent frames. This can be seen in Fig. 7(a) and Fig. 8(a), where since the quality of the motion vectors in the video data of Fig. 8 is better than that of the motion vectors in the video data of Fig. 7, the "accumulated" errors over the extended mosaics in the latter are more obvious than those in the former. However, even with the noticeable motion errors, the generated mosaicking image quality using **ICM** is still comparable with that generated using **DM**, such as Fig. 7(a) and (b). We have evaluated the **ICM** method using all the 10602 frames video data we have used in the evaluation of **LSCA** method, and compared the alignment results with those generated using the **DM** method. Visual inspection shows that there is no or little difference for the generated alignments between the two methods.

Due to the problem of the accumulated error over extended alignment frames, depending on the original quality of the motion vectors, the final visual quality of the generated mosaics may depend on which reference frame is used for mosaic generation. In Figs. 7 and 8, we have shown two examples of the generated alignments using frame 0 and frame 64 as the reference frames, respectively (in (a) and (c)). Since the quality of the motion vectors in the video shot of Fig. 7 is worse than that of the motion vectors in the video shot of Fig. 8, the difference between (a) and (c) in Fig. 7 is larger than that in Fig. 8. However, since the ultimate goal of **ICM** is to develop the tool to allow users to have a quick access to the "background" scene of a video shot, as opposed to developing a tool to generate very-high quality video mosaics, even with the degraded quality of a mosaic such as that in Fig. 7(c), users would still be able to view and then summarize the content of the video data; it is not necessary, nor is our intention, to generate the best-possible visual quality in mosaics for the applications we are addressing.

To show that the quality of **ICM** only using the I-frames, we have run the I-frames only mode of **ICM** for a sequence of 128 I-frames (corresponding to a sequence of 1536 actual frames) to generate the alignments using the two-parameters and six-parameters models, respectively. The results are shown in Fig. 9(a) and (b). These results indicate that if the original data quality is reasonably good, it is sufficient to only use the I-frames to generate the mosaic with acceptable quality. In this example, it has only taken a few seconds to generate these alignments only using the I-frames, which shows that we can further save the computation time to just use the I-frames to generate the mosaics using **ICM**. To further show the time gain quantitatively, we have run the same video sequence with the same models (e.g., the six-parameters model) for both scenarios of using P- and I-frames and using I-frames only, and have recorded the time difference, as shown in Fig. 10. This figure shows that with more frames involved in a sequence for generating an alignment, more time may be saved if only using the I-frames.

6 Conclusions

This project focuses on developing theory and techniques for large scale surveillance video data summarization and unsupervised segmentation of video streams regarding whether there is a presence of independent motion in the streams. We propose a holistic, in-compression approach to efficient video prostanding. By efficient, we mean that the processing speed is close to or even faster than real-time in "normal" platforms (we do not assume using special hardware or any parallel machines) while still maintaining the comparable quality with the state-of-the-art methods. By prostanding, we mean to aim at those tasks that are between the traditional video processing and traditional video understanding. We target surveillance applications. Specifically, we focus on two prostanding tasks: independent motion detection and frame alignment. Solutions developed in the two prostanding tasks provide complementary roles in facilitating efficient browsing of a large collection of surveillance video: the independent motion detection tool allows identifying the shot with the "foreground" target motion without waiting for playing the whole collection of video before identifying the shots; the frame alignment tool allows accessing to the "background" scene immediately through applications such as mosaicking without waiting for playing the whole collection of the video. For the independent motion detection task, we have developed the theory and the technique called **LSCA**. For the frame alignment task, we have developed the theory and the technique called **ICM**. Both techniques are representatives of the holistic, in-compression approach. Theoretical and experimental analyses show that both methods work robustly in solving their problems, and thus demonstrate and validate the holistic, in-compression approach in solving for video prostanding problems.

7 Acknowledgements

We would like to acknowledge the gracious support and the technical advice from Dr. Robert L. Herklotz at AFOSR, the program manager of this project, and Mr. Richard Simard and Dr. Mark A. Robertson at AFRL/IFEC, the AFRL mentors for this project. Stoyan Kurtev of International University – Bremen, Germany, Haroon Khan, Xunyin Wang, and Abhay Garg at SUNY Binghamton, USA, helped in part of the implementation and the experiments for the various versions of the systems developed in this work.

References

[1] G. Adiv. Determining 3D motion and structure from optical flows generated by several moving objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[2] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *IJCV*, 2, 1989.

[3] A.A. Argyros, M.I.A. Lourakis, P.E. Trahanias, and S.C. Orphanoudakis. Fast visual detection of changes in 3D motion. In *Proc. IAPR Workshop on Machine Vision Applications*, 1996.

[4] A.A. Argyros, M.I.A. Lourakis, P.E. Trahanias, and S.C. Orphanoudakis. Independent 3D motion detection through robust regression in depth layers. In *Proc. British Machine Vision Conference*, 1996.

[5] A.A. Argyros, M.I.A. Lourakis, P.E. Trahanias, and S.C. Orphanoudakis. Qualitative detection of 3D motion discontinuities. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[6] A.A. Argyros and S.C. Orphanoudakis. Independent 3D motion detection based on depth elimination in normal flow fields. In *Proc. International Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society Press, 1997.

[7] S. Ayer, P. Schroeter, and J. Bigun. Segmentation of moving objects by robust motion parameter estimation over multiple frames. In *Proc. European Conference on Computer Vision*, 1994.

[8] http://mpeg.telecomitalialab.com/.

[9] P. Bouthemy and E. Francois. Motion segmentation and qualitative dynamic scene analysis from an image sequence. *International Journal of Computer Vision*, 10(2):157–182, 1993.

[10] Q. Cai and J.K. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(11):1241–1247, 1999.

[11] R. Cutler and L.S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):781–796, 2000.

[12] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[13] S. Fejes and L.S. Davis. What can projections of flow fields tell us about the visual motion. In *Proc. International Conf. Computer Vision*, 1998.

[14] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[15] D. Forsyth, J. L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3-D object recognition and pose. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(10):971–991, 1991.

[16] G.H. Golub and C.F.V. Loan. *Matrix Computations, 2nd Ed.* The Johns Hopkins University Press, 1989.

[17] R. Hartley. In defence of the 8-point algorithm. In *Proc. ICCV*. IEEE, 1995.

[18] T.S. Huang and C.H. Lee. Motion and structure from orthographic views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11:536–540, 1989.

[19] M. Irani and P. Anandan. A unified approach to moving object detection in 2D and 3D scenes. In *Proc. of IUW*, 1996.

[20] ISO/IEC 13818-2. *MPEG-2—Information Technology—Generic Coding of Moving Pictures and Associated Audio*, Part 2: Video, 1996.

[21] D.W. Jacobs. *Recognizing 3-D Objects Using 2-D Images*. Ph.D. Dissertation, MIT AI Lab., 1992.

[22] R.C. Jain. Segmentation of frame sequences obtained by a moving observer. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(5):624–629, 1984.

[23] Ryan C. Jones, Daniel DeMenthon, and David S. Doermann. Building mosaics from video using MPEG motion vectors. Technical Report CS-TR-4034, Language and Media Processing Laboratory, University of Maryland, July 1999.

[24] R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: a parallax based approach. In *Proc. International Conf. Pattern Recognition*, 1994.

[25] S. Lang. *Linear Algebra, 3rd Ed.* Springer-Verlag, 1987.

[26] S-W. Lee, Y-M. Kim, and S.W. Choi. Fast scene change detection using direct feature extraction from MPEG compressed videos. *IEEE Trans. Multimedia*, 2(4):240–254, 2000.

[27] M.I.A. Lourakis, A.A. Argyros, and S.C. Orphanoudakis. Independent 3D motion detection using residual parallax normal flow fields. In *Proc. International Conference on Computer Vision*. IEEE Computer Society Press, 1998.

[28] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. DARPA Image Understanding Workshop*, 1981.

[29] Steve Mann and Rosalind W. Picard. Video orbits of the projective group: A simple approach to featureless estimation of parameters. *IEEE Transactions on Image Processing*, 6(9):1281–1295, Sep. 1997.

[30] S. Maybank. *Theory of Reconstruction from Image Motion*. Springer-Verlag, 1993.

[31] Jianhao Meng and Shih-Fu Chang. CVEPS—A compressed video editing and parsing system. In *Proceedings of the ACM International Multimedia Conference and Exhibition*, pages 43–53, 1996.

[32] Ruggero Milanese, Frédéric Deguillaume, and Alain Jacot-Descombes. Video segmentation and camera motion characterization using compressed data. In *Proceedings of SPIE Vol. 3229 – Multimedia Storage and Archiving Systems II*, pages 79–89, 1997.

[33] R.C. Nelson. Qualitative detection of motion by a moving observer. In *Proc. of CVPR*. IEEE, 1991.

[34] Maurizio Pilu. On using raw MPEG motion vectors to determine global camera motion. Technical Report HPL-97-102, Digital Media Department, HP Laboratories Bristol, Aug. 1997.

[35] R. Pless, T. Brodsky, and Y. Aloimonos. Detecting independent motion: the statistics of temporal continuity. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):768–773, 2000.

[36] Richard J. Radke. *Estimation Problems in Digital Video*. PhD thesis, Princeton University, June 2001.

[37] H.S. Sawhney. 3D geometry from planar parallax. In *Proc. International Conference on Computer Vision and Pattern Recognition*, 1994.

[38] H.S. Sawhney, Y. Guo, and R. Kumar. Independent motion detection in 3D scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(10):1191–1199, 2000.

[39] R. Sharma and Y. Aloimonos. Early detection of independent motion from active control of normal image flow patterns. *IEEE Trans. SMC*, 26(1):42–53, 1996.

[40] A. Shashua and N. Navab. Relative affine structure: theory and application to 3D reconstruction from perspective views. In *Proc. International Conference on Computer Vision and Pattern Recognition*, 1994.

[41] J. Shi and C. Tomasi. Good features to track. In *Proc. CVPR*. IEEE, 1994.

[42] S.M. Smith and J.M. Brady. ASSET-2: real time motion segmentation and shape tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(8), 1995.

[43] A.M. Tekalp. *Digital Video Processing*. Prentice Hall, 1995.

[44] W. Thompson, P. Lechleider, and E. Stuck. Detecting moving objects using the rigidity constraint. *PAMI*, 15, 1993.

[45] P.H.S. Torr. Geometric motion segmentation and model selection. *Philosophical Trans. Royal Soc. A*, pages 1321–1340, 1998.

[46] S. Ullman. *The Interpretation of Visual Motion*. MIT Press, 1979.

[47] A. Verri and T. Poggio. Motion field and optical flow: qualitative properties. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(5):490–498, 1989.

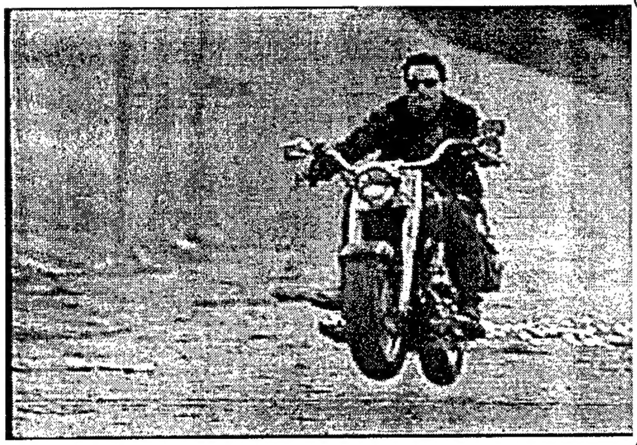
[48] Z. Zhang. *3D Reconstruction under Varying Constraints on Camera Geometry for Robotic Navigation Scenarios*. PhD thesis, CMPSCI 96-08, University of Massachusetts/Amherst, 1996.

[49] Z. Zhang, R. Weiss, and A.R. Hanson. Qualitative obstacle detection. In *Proc. IEEE International Conference on CVPR*. IEEE Computer Society Press, 1994.

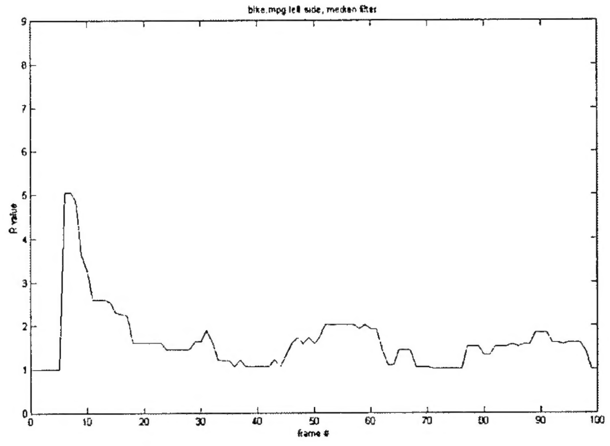
[50] Z. Zhang, R. Weiss, and A.R. Hanson. Obstacle detection based on qualitative and quantitative 3D reconstruction. *PAMI*, 19(1), 1997.

[51] Q. Zheng and R. Chellappa. Automatic feature point extraction and tracking in image sequences for unknown camera motion. In *Proc. ICCV*. IEEE, 1993.

(a)



(b)



(c)

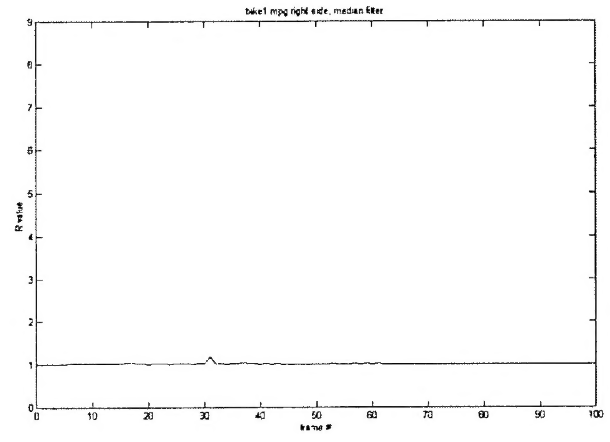


Figure 6: (a) A frame of a movie (100 frames) (b) The R statistics for the left halves of the video (with no independent motion) (c) The R statistics for the right halves of the video (with independent motion).

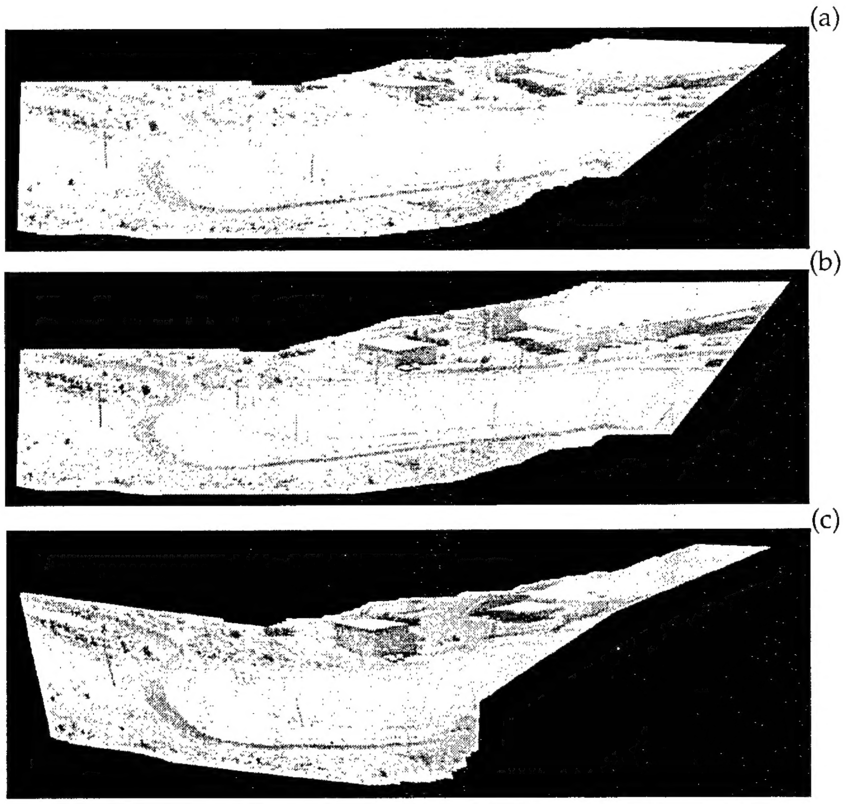


Figure 7: One example of mosaicking a 128 frames shot (a) Generated mosaicking image using the **ICM** frame alignment method with frame 0 as the reference frame (b) Generated mosaicking image using the **DM** frame alignment method with frame 0 as the reference frame (c) Generated mosaicking image using the **ICM** frame alignment method with frame 64 as the reference frame.

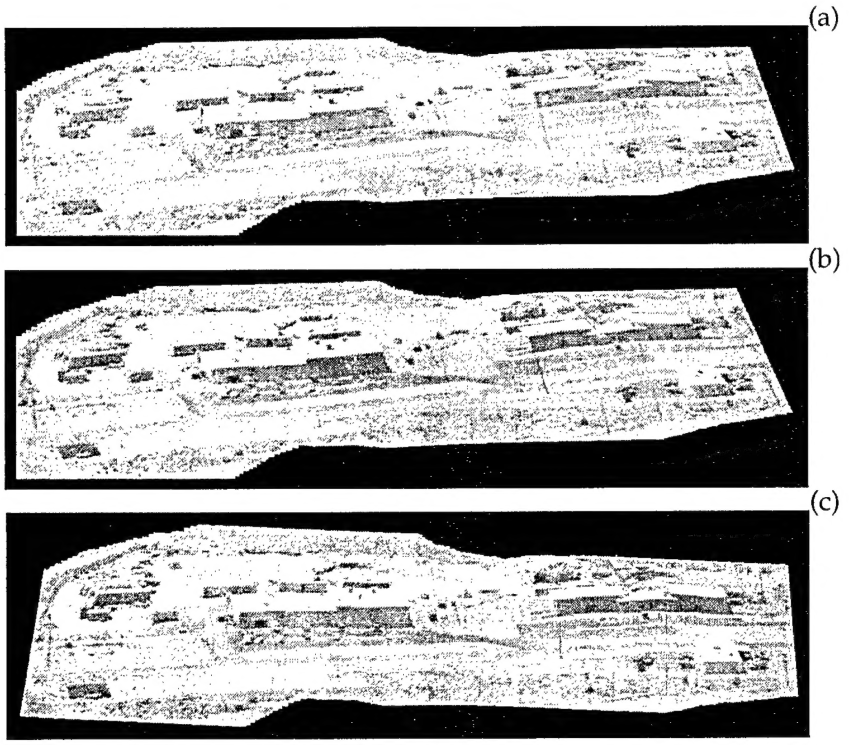


Figure 8: Another example of mosaicking a 128 frames shot (a) Generated mosaicking image using the **ICM** frame alignment method with frame 0 as the reference frame (b) Generated mosaicking image using the **DM** frame alignment method with frame 0 as the reference frame (c) Generated mosaicking image using the **ICM** frame alignment method with frame 64 as the reference frame.

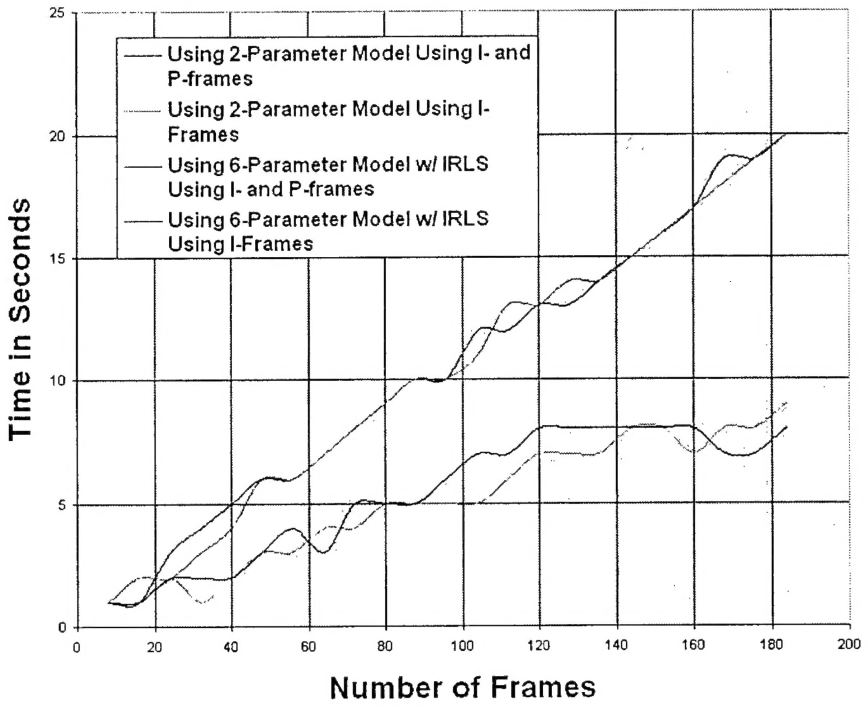


Figure 10: The relationship between the time saved for generating the alignments and the number of frames involved in the video sequence when the two scenarios of using both I- and P-frames and using I-frames only are compared.